SIENA*college*
Computer Science

Computer Science 340
Programming Languages
Siena College
Fall 2023

Lab 1: Python Basics for Java Programmers

Due: 11:59 PM, Tuesday, September 12, 2023

In this lab, you will experiment with some of the features of the Python programming language, focusing on things that it does differently from Java.

This is an individual lab, but you are welcome to collaborate with classmates as you work through. However, the work you submit must ultimately be your own, and you are responsible for understanding everything in the lab.

Learning goals:

1. Learn the basics of the Python programming language and how it differs from Java.

Getting Set Up

In Canvas, you will find a link to follow to set up your GitHub repository, which will be named `python-lab-yourgitname`, for this lab.

Developing and Running Python Programs

You may develop and run the Python programs for this class in whatever way is most convenient to you. You should make sure you have a fairly recent version of Python. As of this writing, the current version is 3.11.5. Anything about 3.8 or newer should be sufficient.

Some options for running Python:

- Within VS Code with Python extensions installed.
- Within a GitHub Codespace.
- On noreaster as `python3`.

We will look at each of these options briefly in class.

As long as you can work with your programs in your repository (or put them back there when completed), you should be fine.

You can use Python interactively through via REPL (read-evaluate-print-loop) interface or have it run complete programs that you write in files. This flexibility comes from the fact that Python is an *interpreted language* (as opposed to a *compiled language* like C or Java).

As a group, we will read and try out (using the REPL interface) the examples in Section 3 of the Python Tutorial.

Console Input/Output

Unlike Java, you don't need to have a class or even a method to have a Python program. A "Hello, World!" program here is just a one-liner using the `print` function.

Your repository includes such a program called `hello.py`. Make sure you can run this program. Don't run this one in the REPL interface, instead use a command like the following from a command prompt:

```
python3 hello.py
```

> **Practice Program:** Modify the program to print a second message, and to have a comment at the top with your name. (2 points)

Also unlike Java, Python makes it simple to read input from the keyboard. The builtin function `input`, which takes an optional string parameter that will be presented as a prompt, returns a line of text entered at the keyboard.

> **Practice Program:** Create a new program called `hellome.py` that prompts for your name then greets you by name. (3 points)

With this program, experiment with different ways of printing the message (comma-separated parameters to the `print` function, Java-style string concatenation) to get the desired output:

```
$ python3 hellome.py
What's your name? Roger
Hello, Roger!
```

Conditionals and Loops

Read Sections 4.1-4.4 of the Python tutorial and try out some of the examples.

One very important thing to be aware of once we start having control structures in our Python programs is that there are no curly braces to delimit blocks of code – the code that forms the body of an `if` statement or a `while` loop is determined by indentation!

> **Practice Program:** Create a new program called `hellomore.py` that first prompts for your name. If your name has no more than 5 letters, it prints a message that your name is pretty short, then repeats a greeting a number of times equal to the number of letters in your name. If your name has 6 or more characters, if prints a message that your name is long and greets you just once. The `len` function should come in handy. (10 points)

Sample runs:

```
$ python3 hellotimes.py
What's your name? Roger
You have a short name, I'll greet you once per letter!
Hello, Roger!
Hello, Roger!
Hello, Roger!
Hello, Roger!
Hello, Roger!
$ python3 hellotimes.py
What's your name? BillyJoeBob
That's a pretty long name, I will greet you just once.
Hello, BillyJoeBob!
```

More Input

One mechanism to handle more complicated input in Python is to parse the string returned by the `input` function, converting types as needed. For example, if you have the line of input

```
cloudy 75 .13
```

which is intended to represent an hourly weather observation, with the sky conditions, temperature in Fahrenheit, and inches of precipitation since the last observation.

The following Python statements will read that as a line of input, split it into an array based on a space as a delimiter (the default), then store each value in an appropriate variable, converting as needed.

```
line = input()
fields = line.split()
cond = fields[0]
temp = int(fields[1])
precip = float(fields[2])
```

Practice Program: Write the program described below in a file `grades.py` using what you've learned in this lab. (25 points)

Your program will read in the names and grades for a group of students, then report some statistics about those grades. The statistics to be gathered are the number of scores entered, the highest score and the name(s) of the student(s) who earned that score, the lowest score and the name(s) of the student(s) who earned that score, and the average score for the class.

So if the names and scores are as follows:

```
Luke 76
Hermione 99
Wally 34
Linus 99
Penny 25
```

then your program would print output:

```
There were 5 scores, averaging 66.6.
The lowest score was 25 by Penny.
The highest score was 99 by Hermione and Linus.
```

- The program will need loops in two places: (i) to be able to read in data repeatedly, and (ii) to ensure that any erroneous input is detected and re-read until a valid input is obtained.
- The program should read in names (single-word names are sufficient here) and scores until the name is entered as `done`, at which time we stop reading names and scores and report the final statistics.
- All scores should be in a valid range 0 to 100. If a score is entered outside of that range, your program should issue an appropriate message and re-read only the score (not the name, we already know that!) until a valid score is entered.
- If there is a tie (among any number of students) for the highest and/or lowest score, all students who tied the high or low should have their name included in the printouts at the end.
- If no valid names and scores are entered, a special message "`You did not enter any scores!`" should be printed rather than potentially erroneous stats.

Submission

Commit and push!

Grading

This assignment will be graded out of 40 points.

Feature	Value	Score
`hello.py` modification	2	
`hellome.py`	3	
`hellomore.py`	10	
`grades.py`	25	
Total	40	